LOADSTAR LETTER #65

Nate Dannenberg Designs Quick-Scan64 Interface for the Connectix QuickCam

By Jeff Jones. Want to take live video with your Commodore 64 or 128? Well, if you have a SuperCPU (needed only for the high-res modes), you can use the QuickScan 64 interface to hook up a Connectix QuickCam (B&W Model for now) to your Commodore and have at it! One of the most recurring questions asked on the C-64 platform is, "Can you do so-and-so on a C-64?" the answer has always been, "Sure, if someone develops xxx for the C-64." Well, Nate Dannenberg has pulled out his tweezers and soldering iron in order to design a video interface for the C-64. The video camera that it's designed for can be obtained inexpensively so if you're a hardware hack, this may be a project for you. Check out the interview with Nate inside!

jeff@LOADSTAR.com
US MAIL: ATTN. Jeff Jones
J & F Publishing
P.O. Box 30008 Shreveport
LA 71130-0008
Phone: 318/221-8718,
Fax: 318/221-8870

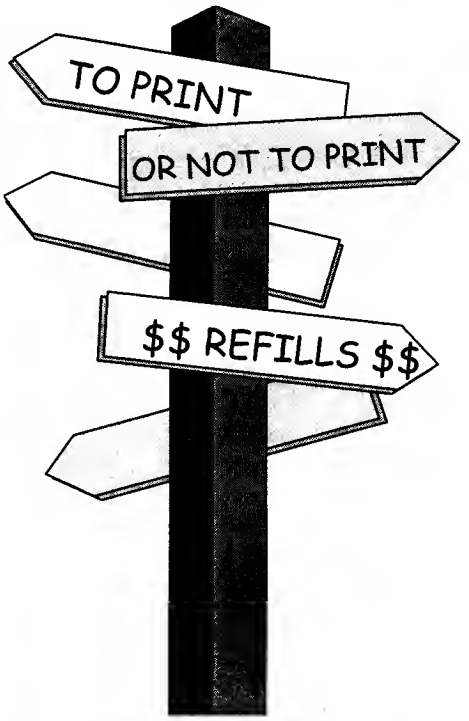
Inserting Graphics And Printing Your Documents

By Jeff Jones. Somewhere out there, there are a million or so needless pixels being marked to paper. Someone liked an image on his screen so much that they have decided to print it out. Rarely does an image look better on paper than it does on the screen so why print it unless there's a specific purpose?

We know there are print junkies out there because throughout the years, Loadstar's print bugs have been the most quickly and most often reported. That's because there's too much printing going on! Printing isn't fun to me. I consider it an expense — a task — a *job*. This kind of thinking saves me money. For me printing is only fun when you buy a new printer and you say, "Wow! Nice print!" After that why print when you don't have to? And when you do print, why print graphics when you don't have to?

There is a reason to avoid graphics. In general graphics eat up ink and toner. Light-colored graphics eat less toner and ink, but most graphics just plain ain't light. As black as this page seems, most of it is white. If I plopped a gratuitous graphic, I use a bit more ink. The image of Nate Dannenberg, though smaller in area than the image on the right, uses up much more ink.

But we need graphics. I don't use many graphics in the Loadstar Letter because we rarely use film anymore, which makes for the best looking pages. But in general a publication like a newsletter *needs* graphics just to break up the monotony of text. Then when you



add one graphic, it's considered good to balance it out with another somewhere else on a page. It's a psychological thing. Some people see a sea of text with no charts, graphics, pull-quotes or sidebars, and they consider it more work to read than the same text with graphics. Even most "text books" look like newsletters inside. This rule doesn't apply to personal correspondence, yet I often receive letters riddled with

"Jeff Becomes a Gamester And Prognosticator"

This Christmas I purchased a Playstation for my daughters. I think the Playstation has done for computing what the C-64 and Amigas did in the 80s. Here you have this little $200 machine with better graphics performance than "real" computers. I must say that I was truly amazed when I saw the little machine in action, particularly when playing Soul Blade. I can play this game all day, and my daughters both beat the heck out of me since they have more access (read practice) to the games. The opening animation sequence is better than some Disney animation I've seen. Within the game itself, some of the animation is so fluid that I can see cartoon software being written for television producers. I'm not kidding. Perhaps my 200-MHz Pentium MMX PC with 80 MB RAM and 2 MB graphics accelerator is better on paper, but the Playstation sure seems to perform better. Most any decent on-the-fly animation I've seen on my PC has been in a small window. The Playstation is full-screen. I will admit that I've produced photo-realistic sharper-than-cinematic full-screen morphs, but again, much of the action I'm seeing is better than some (but not all) cartoons on television. Given that the characters are generated in real time, I can see near-photo realism if they allowed for rendering time as they do in the fantastic opening animation, again better than most Disney. I foresee three or four people with controllers, one controlling eyes, the other, lips, and others controlling body motion. This is what we're currently doing with animatronics. I believe Sigourney Weaver's alien queen had twelve people controlling it.

At times like this, I wish I were in the big, big ***big*** software industry. I would design an animator's pack that uses multi-tracking. In this way a single animator could control facial, body and other attributes of a single or many characters with a single controller, and then go back over and over the sequence until it's perfect. Next he can render it, which might take minutes to hours, and then record to video. This type of software sort of exists now, but as far as I know, the gaming method of control hasn't been implemented in professional programs. It's more script, pre-defined and drag oriented. Only the military seems to have discovered that if you want to control something complex, you need a game pad and arcade feel. All my adult life, I've wanted to do videos. I mean real productions. Perhaps in the near future, I can.

What's With The Font Size?

Last issue the font size increased from 10-point to 12-point because of complaints of fine print. I've received a lot of Email about this. So now the font is larger. I've also received input that the margin should be increased so that the newsletter can be cleanly inserted into a spiral binder. No can do for now, especially with the larger font. It would cut down too much on content.

(Continued from page 1)

graphics — sometimes even *color* graphics.

Nothing has changed in the print industry. Last time I heard, printers were still expensive to use, particularly ink jets, which have started a new and profitable ink industry. Whenever a new printer is reviewed, particularly color printers, the subject of *cost per page* comes up. Printing costs *money*. Color ink not only costs more, but the ink and/or toner seem to be used up much faster than black or other monochrome media.

In the old days, I only printed when I wrote letters or when I had to list a program for debugging. I enjoyed very low print costs. When I started editing this newsletter, I began printing more and more. Besides the master copy of the newsletter, I inevitably used to print at least two other newsletters because of mistakes

Available from LOADSTAR!

Chris Abbot's goal was to professionally reproduce well-loved Commodore demo and game tunes. He pulls this off quite well, using state-of-the-art MIDI equipment. These CDs were not manufactured on a PC's CD recorder. They were professionally pressed, fully packaged and contain a nice little booklet with explanations for each song along with a Rob Hubbard interview. You should get this CD, if only as a collector's item. The item number is #200122 $20.00

LOADSTAR 1-800-594-3370

and revisions. This translates to 280 pages printed per year just for the Loadstar Letter.

Because of the Loadstar Letter, I purchased an ink jet printer, the Canon BJ-5 (I think), which was a piece of junk. Within a few months I spent more than $70 in black ink. Until that time I spent perhaps a total of $30 in ten years for ink or ribbons. I didn't own a color printer until 1995. In 1995 I spent way over $100 in ink — perhaps even $200.

Some might feel that it's none of my business, but I cringe when I see gratuitous graphics in a letter. I think of the cost to the sender, particularly when it's a color print of a Loadstar graphic that I've seen anyway.

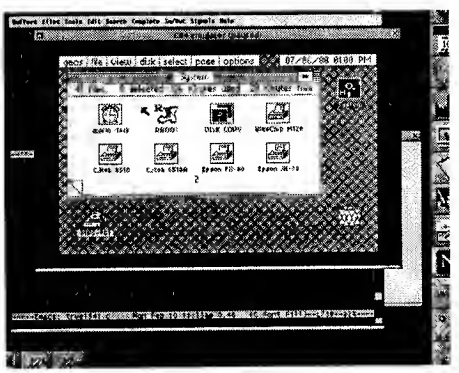
Despite my seeming anti-print crusade, I print more than most people. Sometimes I print out of laziness. I'll type down a name and address or phone number and might print out a whole page with only three lines on it. My wife tells me I'm wasting paper. I just tell her that my reams of laser pages cost less than the kids' lined paper. Moreover a few lines of ink or toner is nothing, particularly when you're printing text.

Beyond my snippets, I print at least two versions of this newsletter, which amounts to thirty pages per month, plus many high-density samples (pictures that use up a ton of toner) for my various customers. Then there are letters and web pages that I might print out for my own reference. All in all, I might print 100 or more pages per month, yet I haven't had to replace my toner for an entire year. Mind you my toner only lasted two months when I first purchased my laser printer. That was because I printed 250 four-page wedding programs with plenty of black. It also could have been because my original toner cartridge wasn't as filled as my recharged cartridge.

Here are a few guidelines I use for printing that have saved me money.

- Printing does have its place, particularly for proofreading
- Resist the urge to print a high-density page just for the pleasure of seeing it.
- Print nothing before you're ready to proofread it. In other words printing is a dress rehearsal.
- Large blocks of black will use up more ink and cost you more money. They might also fail to reproduce on a bad photocopier.

New Vice Release



By Andreas Boose. Version 0.16.0 of VICE, the Versatile Commodore Emulator, has been released. VICE emulates the Commodore C64, C128, VIC20, PET 8-bit computers and the CBM-II (AKA C610) and is completely written in C/C++; it runs on MS-DOS and Unix systems.

VICE is a *Versatile Commodore Emulator*, i.e. a program that runs on a Unix or MS-DOS/Windows 95 machine and executes programs intended for the old 8-bit Commodore computers. The current version emulates the C64, the C128 (save the 80 columns screen), the VIC20, all the PET models (except the SuperPET 9000, which is out of line anyway) and the CBM-II (AKA C610).

For the first time the MS Windows 32-bit port of VICE is released. This port is brand new and has not reached full quality of the Unix and MS-DOS versions yet. Therefore it is tagged as "ALPHA" release.

System requirements for WinVICE 0.16 ALPHA:

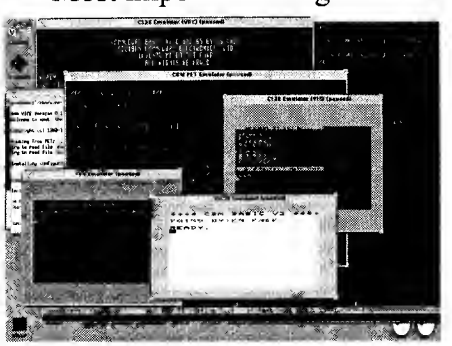
- MS Windows 95 and DirectX installed (5.0 or higher recommended)
- MS Windows 98
- MS Windows NT4.0 and SP3 (or higher) installed

Limitations of this alpha release:

- Some functionality is missing. (Where is the joystick emulation?)
- Palette changes can lead to wrong colors. This can be avoided by using 16/24/32 bit RGB color depths.

VICE is free software released under the GNU General Public License, and as such it comes with full source code.

Most important changes since



version 0.16 include:

Snapshot functionality: It is now possible to save the emulator's state into a file, and load it back at any time. This works even with disk emulation turned on, and can include ROMs and Disk images. Snapshot files can be moved across all platforms supported by VICE.

C64 changes:

- A couple of bugs in the emulation of 6510 CPU core, VIC-II and REU have been fixed.
- Final Cartridge, Ocean Cartridge, Super Snapshot 4 and 16 KB CRT image support has been added.
- Display modes in Ultimax mode have been corrected.

C128 changes: Some memory banking bugs have been fixed (C128 GEOS works).

VIC20 changes: RS232 interface has been added.

CBM-II changes: A new Commodore 610 (AKA CBM-II) emulator has been added.

Drive changes:

- Drive 9 can be emulated at hardware level.
- The 1571, 1581 and 2031 disk drives are emulated.
- Moved drive ROM images into a separate directory.
- Emulation of drives has been sped up consuming less host CPU power.

SID changes

- Dag Lem's enhanced reSID engine is now part of VICE.
- A bug in the random number generator has been fixed.

Unix changes

- The "Custom" joystick mapping is now compatible with the German keyboard layout, and possibly other non-US layouts.
- Non-default screen depths should really work on X11 now.
- Generation of core dumps can be controlled from command line.
- Smart attaching lynx and zipcode images works correctly now.

MS-DOS changes

- The menu system now uses a CBM-lookalike character set borrowed from Star Commander.
- You can now browse disk images both with the graphics and business CBM character sets, and you can autostart any file on a disk image.
- A file descriptor leak has been fixed.

C1541 changes

- C1541 has been restructured; the batch mode syntax has changed, and some commands take different parameters.
- You can create D71 and D81 disk images now.
- Lynx and zipcode support has been improved.

Miscellaneous changes

- A new logging system has been added.
- The Monitor now handles zero page watch points triggered by memory wrap-around too.
- Two new palette entries have been added.

The VICE 0.16.0 source archive is available at:

ftp://ftp.funet.fi/pub/cbm/crossplatform/emulators/VICE/vice-0.16.0.tar.gz

The VICE 0.16.0 MS-DOS binary archive is available at:

ftp://ftp.funet.fi/pub/cbm/crossplatform/emulators/VICE/vice0160.zip

The VICE 0.16.0 WIN32 binary archive is available at:

ftp://ftp.funet.fi/pub/cbm/crossplatform/emulators/VICE/WinVICE-0.16.0.zip

For more information, check out the new VICE home page at:

http://www.cs.cmu.edu/~dsladic/vice/vice.html

MfG Andreas

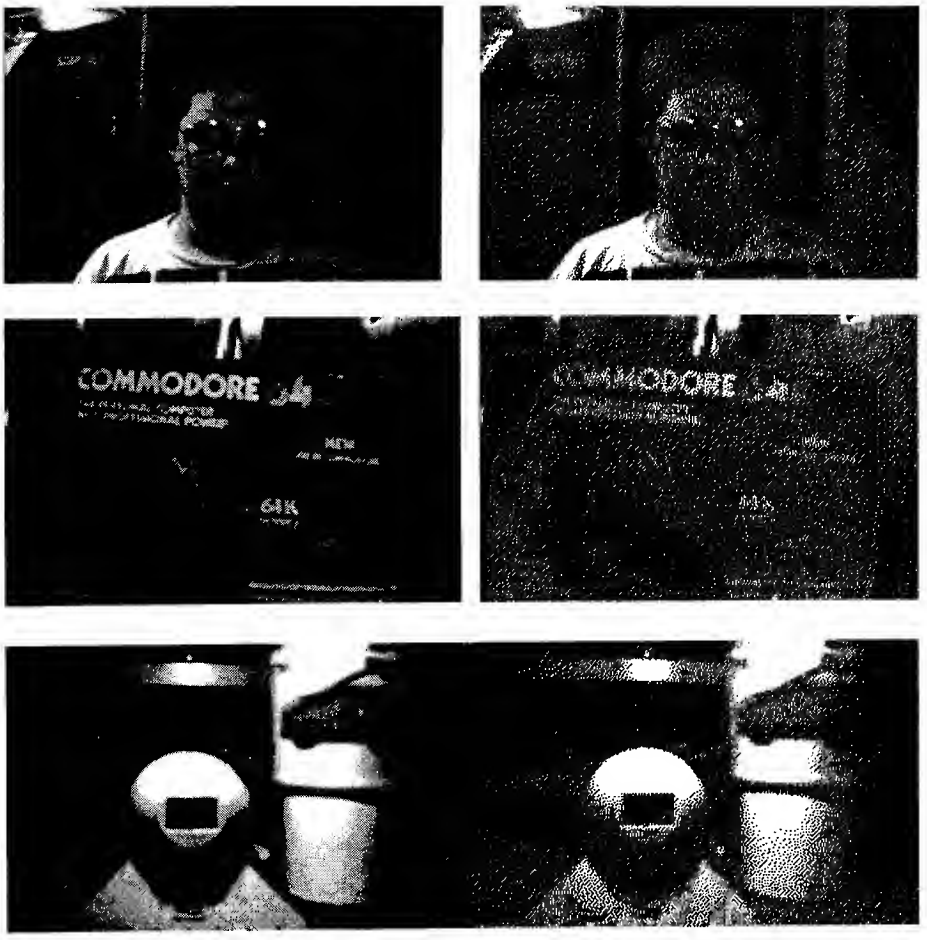
An Interview With Nate Dannenberg

Jeff: Please introduce yourself to our readers and list your accomplishments.

Nate: My name is Nate Dannenberg. Born in 1974 in

ACTUAL OUTPUT

Quickcam Image | Actual C-64 Screen



Chanute Kansas, I moved around the state with my family looking for that "perfect little spot" to live. 15 years ago we settled in Mulvane, KS. About three years after that, I got my first Commodore 64.

[Head hangs low] I'm a Commodore 64 addict. :-)

To date I've written many programs. The most notable are Sound Studio 128, which plays PC *.WAV files and can play or record Commodore *.RAW files, and Modplay 64/128, which plays Amiga/Protracker *.MOD files.

Jeff: Please explain your Quikcam interface and software system. What does it do and what doesn't it do?

Nate: Well for starters, it's a very simple interface based on the 4066 quad analog switch chip. The idea was to give the user port enough control lines to talk to the camera, which is what the 4066 does.

The software does only one thing, and does it well - whether or not you have a Super CPU, the programs will allow you to read, in real-time, live video straight from the camera.

There are four modes available:

- 80x50 in 4 grays
- 160x100 in 4 grays
- 320x200 mono, with 16-level Floyd-Steinberg dithering
- 320x200 in 16 grays. MCM-Lace

The first three work on any C64 or C128, Super CPU or not. The MCMLace mode, which must render as the image is read in, requires a SuperCPU or else the images become washed out.

Jeff: What is your electronic background?

Nate: I've been doodling around with various electronics for about 12 years; pretty much as long as I've had a Commodore computer.

I've built the simple stuff - radios, amplifiers and the like - in fact my first electronics kit was Radio Shack's original 100-in-1 kit.

After an unsuccessful attempt to put stereo SIDs into a C64C, I finally caught on and learned how to solder properly (prior to that, everything was wire-wrapped or spliced, with the exception of this stereo SID attempt).

It wasn't long after when I built my first external C64 interface, which was just a simple RS232 cable. I began making these for friends, and finally graduated to printer circuit boards.

Since then I've been making all sorts of small devices for the C64. My website tells all about the items I still make.

Jeff: What made you decide

to take on this project?

Well, sometime ago, Robin Harbron (Macbeth/PSW) and I got to talking about making an Internet "phone" program. Of course, most such programs today offer video capabilities as well as audio.

So I got to thinking - how much memory and processor power does it take to handle a simple 80x50 image? Well, not much of course - we've all seen 80x50 plasmas in demos, and these are even more complex than the display mode I had in mind.

So, I contacted Connectix and requested interfacing and programming details on the QuickCam - the only camera I considered to be "worthwhile" at the time (it was, after all, the most popular). Soon after, I received the information I wanted and began memorizing the timing diagrams and hardware aspect of the camera.

Eventually I saved up enough extra cash to purchase a black and white camera (it was $99 at the time), and built an interface to connect it.

After further study of the software protocol I got my first image from the camera. Mind you it was only a test image, which isn't what I was trying to get, but it proved the camera could send data to the computer.

The test image was a result of a couple of miswired connections on the interface, which was easy to fix. Like magic, live video started to appear.

Now that it works, I plan to use it for AVLink - the Internet videophone program I want to write.

Jeff: Are you afraid of damaging your C-64 or external hardware when you take on projects like these? Any horror stories?

Nate: I'm always afraid of damaging something in the computer when I start a project like this, no doubt of that.

I can recall only one horror story, from at least 10 years ago. I was trying to install those stereo SIDs into a C64C. Everything was going fine until I dripped a little bit of solder onto the motherboard. Not knowing to just let it cool first, I tried to draw it off with the soldering iron. Next thing I know I have an inch-wide burned-off spot on the board - all of the traces in that area had burned or worn off from my attempts to remove the solder.

Of course I've had my share of blown CIA's and SID chips trying to interface this or that to my computer. In fact, the first version of the QuickScan interface killed a CIA chip, which I wasn't ever able to replace. Instead, I opted to swap out the motherboard of the computer - I wasn't about to try to de-solder a 48-pin CIA chip :-)

That first version blew the CIA chip because of excessive current draw - I was using 2N2222 transistors, driving too many off one line.

Jeff: How much does the Quick cam cost and where can one obtain one?

Nate: Well sources tell me you can get the black and white model for as low as $50 new, from various computer retailers. A friend of mine has a source of them for $25 apiece. The B&W model has already been discontinued, but there are still plenty of them to be had.

The color Quickcam is about $200 right now, and can be found at any computer store. Eventually, when I get one of these, I'll modify the interface (if needed) to work with one. Adrian Gonzalez (_dW/Style) has plans to write a 320x200 136-color IFLI renderer for the new camera.

He says the renderer will be quite slow even on a Super CPU, but that's because it will be dealing with truecolor (24-bit) data and it will be a true renderer in the official sense of the word - optimize the image to display 100% accurately on the display mode in use.

Jeff: How do you see Commodore users using your system?

Nate: Well to be honest, the intent of the system is to eventually let a Commodore user talk with another Commodore use using both audio and video, and that's exactly where this camera comes in - I want to use it to supply the video part of the program, just like these high-on-the-horse PC's out there.

The audio would be supplied with a simple ADC chip, and everything will be connected to the expansion port, as part of the "Generation 2" interface I am designing.

Of course, users can also take live-video movies as well. The Generation 1 interface is a bus of a bus hog, but with this much data that doesn't really matter. Movies could be taken and stored in expansion RAM (i.e. an REU or SuperRAM memory) and saved to disk after editing, clipping, or whatever else the software will do. I haven't written anything to save these images yet, but that's

just a few days worth of work to make a full fledged program.

Jeff: Your Quick cam interface and software work with a SuperCPU-equipped C-64 in 128 mode. Could you explain why a SuperCPU is required, and why the 64 mode?

The 64 mode was chosen because the SuperCPU 64 is the only model I have.

The SuperCPU isn't required at all by the program in any way, at least not exactly...

The camera has a design flaw in it (as most of these types of cameras do). If the camera exposes an image to its light-sensitive image surface (A CCD array to be exact), it then transfers the captured image to an on-board buffer.

That buffer is where the flaw exists - it is a "capacitive storage buffer" as the spec sheets put it, and is leaky. That means that after a certain amount of time (about three seconds), the image becomes washed out.

The documentation says this washout problem is supposed to be auto correcting (the camera is supposed to automatically expose another frame when the image starts to fade), but this doesn't seem to be the case when an image is being downloaded.

The problem exists for the C64 in only one video mode - 320x200 16-gray MCMLace. The problem here is that there simply is not enough memory available in the C64 to download the entire image and then render it.

Therefore, the image is rendered as it is downloaded. This means the camera is transferring data while at the same time, the image is becoming more and more washed out.

Approximate SuperCPU scan times (Stock time in seconds)

1 - 80x50 in 4 grays 0.5 30 Frames/sec (est.)
2 - 160x100 in 4 grays 1.75 6 Frames/sec
3 - 320x200 16-level F/S 17 1 Frame/sec
4 - 320x200 16-gray MCMLace 18 1 Frame/sec

The other three modes download entire frames at a time before rendering them; hence they work perfectly even on a stock C64.

Jeff: How long does it take to download an image through the user port?

Nate: It takes about 6 seconds to download a 320x200 frame from the camera, and another 11 seconds to render it in mode 3 above, on a stock C64. Since Mode 4 (MCM-Lace) renders as it downloads due to memory limitations, this mode is measured only as 18 seconds for the entire frame, with no "download" time.

Mode 4 really needs a Super CPU; otherwise images returned in that mode look washed out.

Jeff: Anything concrete yet on the specs for the generation 2?

Nate: Only that it will use the Expansion port and will be able to work with 16-bit Super CPU accesses. It will have at least an ADC chip onboard as well. A Passthru port will be included to make it work with other expansion port hardware.

It will require a jumper to run inside the C64 to pick up the I/O enable line, so that the card will disappear from the memory map when I/O devices are switched out.

There is a possibility that this card may be available as an add-on to the PowerSID 4000 add-on board, in which case it need only contain the video portion of the circuit. Audio would be supplied by the PowerSID 4000's onboard hardware. No extra jumpers would be needed, as the PowerSID has onboard addressing circuitry (and it's own set of jumper wires).

http://www2.southwind.net/~natedac/

Some Of Nate's Other Projects

All these products can be found at Nate's Webpage: http://www2.southwind.net/~natedac/

Some stuff is still being developed, while other items are now available.

Some items were originally being sold through Arkanix Labs, but due to their closure of Commodore support in November of 1997, I've gone on my own with everything. All of the hardware I make is bare-board hardware. There are no cases used here, as that is cost-prohibitive.

A simple RS232 interface

Need an RS232 adapter? Here is a simple interface that I used to use for a long time. It worked with every modem I ever tried it with and is good to 9600 BPS.

The 8BSS

This is an 8-bit stereo sampler board. This little device supplies two National Semiconductor ADC0820 chips, and plugs into your User Port to deliver up to 100 kHz 8-bit digital input. Actual recording speeds will be determined by the software in use and the speed of the user's computer. Price: $25
Board Layouts/Parts list available by Email.

The DigiMax

This little device is pretty simple. It contains a single Maxim MAX505 chip, which provides for four channels of sound at 8 bit resolution, at up to some 250 kHz, depending on the user's computer speed and software settings. Like the 8BSS, this board plugs into your User Port. Price: $20.
Schematics available in DisC=overy issue #1.

A Simple R-2R ladder

Don't have an 8-bit DAC for use with Modplay? Don't have the time or money to build or buy a Digimax? Build your own 8-bit DAC then! Here is a simple "R-2R" ladder, which works well, especially if you use it with a latched parallel port (like the C64 user port).

DualSID

This is a plug-in internal stereo SID expander board for the C64 or C128. It requires two SID chips, which normally would consist of your stock SID chip plus one extra that you must supply. The board includes a disable switch and output jack, which must be mounted to the back or side of the computer. Addressable at $D700 (C128 only), $DE00, or $DF00. Price: $20. With one SID chip: $35
Board design is (P) and (C) 1996 by Thorsten Oelfke.

The following are various incarnations of the PowerSID board designed by Shaun Halstead (Tesla/DAC) and me. You can contact Shaun by sending Email to [tesla@southwind.net]. The PowerSID is a multi-chip device that allows a single C64 to play music in more than 6 voices (up to 27), without so much as an accelerator or another computer. These PowerSID boards are in the design stage, and have not yet been prototyped. PowerSID is a trademark of Digital Audio Concepts.

PowerSID 1200

This is the basic model of the PowerSID system. It starts out with a simple four-SID expander board for the C64. Connected to your expansion port, it can be addressed at $DE00 or $DF00 on the C64 or C128. C128 users may also place the board at $D700 using the included clip-on jumper wire, which must be run to a chip inside the computer. Price: $50. With four SID chips: $110

PowerSID 2400

This is an add-on board for the PowerSID 1200, which brings the total SID chip count to 8, for a total of 24 added voices. Price: $35. With four SID chips: $95

PowerSID 4000

This is another add-on board for the PowerSID 1200, which supercedes the PowerSID 2400 add-on board. Like the PowerSID 2400 board, this one supplies up to four additional SID chips. In addition, this board sports two ADC0820 chips and one MAX505 chip, to provide two 8-bit digital inputs and four 8-bit digital outputs.

A jumper is supplied to allow the unit to disable the $D038-$D03F address range (which is where the ADC's and DAC's sit), while I/O devices are switched out. This is in addition to any jumper used for $D700 addressing, if used on a C128. Price: $50. With four SID chips: $110

As far as Shaun and I can tell, all of the above projects will work with most CMD hardware. In some cases a C128 might be required due to addressing conflicts (for example, when using the RamLink, or certain utility cartridges). All of these projects should work with a Super CPU.

The projected costs shown here are examples only, and are not guaranteed to be correct when production time rolls around. The projected costs of boards that include SID chips are based on an estimated price of $15 per SID chip. Actual cost may be higher or lower depending on what source we end up using, to supply SID chips for these boards. Unless other arrangements are made (or the supply dries up), matched sets of MOS6581 SID chips will be supplied.

All prices and specifications are subject to change without notice.

CMD, RamLink, and Super CPU are trademarks of Creative Micro Designs, Inc. Digital Audio Concepts is in no way affiliated with CMD, but I happen to like their hardware. So there. :-)

Stephen Judd's New El Cheapo Assembler

By Robin Harbron. Ever wished you could write some code like:

```
LDX #60000
loop DEX
BNE loop
```

How about changing both the border and background color with a single STA?

```
LDA #$1112
STA 53280
```

Ever missed these instructions?

```
PHX, PHY, PLX, PLY
STX blah,Y
INC A, DEC A
BRA (branch always)
TXY
```

Well, now you can make use of all of these features – with the help of CMD's SuperCPU and Steve Judd's new powerhouse combination of tools – El Cheapo Assembler (ECA) and JamaicaMON 2 (Jammon). Jammon has been around for about a year now, but version 2 was just finished a month or so ago. It made C= history as the first 65816-aware machine language monitor. It's also inherited a bit of a legacy as well, as the code for Jammon is largely based on the original code for SuperMON made by Jim Butterfield over 15 years ago. SuperMON was geared to the 8-bit 65xx series of processors found in the PET, VIC20 and C64. Jammon takes this a step further, and adds the 16-bit 65816 instructions found in the SuperCPU to its vocabulary.

Jammon is an exciting tool, but not very practical for developing the large programs that will be able to harness the SuperCPU's power. I'll let Steve Judd's public announcement of ECA speak for a moment:

Subject: 65816 Assembler -- Public Beta Release
Date: 3 Jan 1999 20:18:35 GMT
From: judd@merle.acns.nwu.edu (Stephen Judd)
Organization: Northwestern University, Evanston, IL
Newsgroups: comp.sys.cbm

Season's Greetings,

Because I am moving across country and won't get to work on it for a while, I am releasing El Cheapo Assembler out into the world:

http://stratus.esam.nwu.edu/~judd/fridge/cheapo/

Some features of CheapAss include:

- *65816 opcodes*
- *Integrated editor/assembler/ML monitor*
- *Useful pseudo-opcodes for conditional assembly, storage, etc.*

Requirements:

- SuperCPU (and a C64!)

Actually it will work without a SuperCPU, but you cannot save any object code (or use the monitor). The current program has not been tested thoroughly, and may have some bugs left in it. Nevertheless, I have successfully assembled an earlier version of El Cheapo Assembler in El Cheapo Assembler.

Thanks, and see you all in a few weeks!

-Steve

Steve, being the humble fellow he is, doesn't mention that he actually wrote ECA in just a few weeks, in the middle of finishing off his Ph.D. in Applied Mathematics, packing up all his belongings, defending his thesis, obtaining his degree, and moving across the country to his new job that only Ph.D.-types can understand, never mind actually get hired for.

I'd like to point out that this program is much more important than most other programs. It's a long awaited tool that will allow other avid SuperCPU programmers to finally create some of the programs we've long been promised. CMD hasn't managed to get an assembler to us in the 2.5 years that the SuperCPU has been around. Although to be fair, in all I've heard, it sounds like fate has conspired against the release of an assembler for the SuperCPU. Two different assemblers were apparently "almost done" when the authors dropped out of the C= world for various reasons. I'm glad fate has spared Steve thus far.

A short aside: In North America, the commonly accepted abbreviation for assembler is ASM. Over in Germany, unhindered by the bad connotation in English, they use

the more logical abbreviation of ASS. This has led to the perhaps unfortunate, and certainly humorous naming of several assemblers: TurboASS, SuperASS, and what I believe to be the first native SuperCPU assembler: AssBlaster.

Unfortunately, AssBlaster was neither very friendly nor usable program, and just never caught on over here. Perhaps this little aside will give you a better as to why Steve named his program El Cheapo Assembler.

I've spent a number of hours in ECA so far, and written Steve about possible improvements and bug fixes. The most exciting thing about this program is that it is supported by someone who takes pride in his work, and is always willing to listen to people who use the program.

I found ECA remarkably easy to use. The built in editor has most every feature I could think of. It's a full screen editor, which means you can simply scroll around your source code, rather than having to cope with any of those BASIC based editors – or even worse, those dreaded line editors. Formatting is automatically done for you so you simply type in your line – label, opcode and comment – and everything is automatically aligned in columns.

A very nice feature is the way labels are implemented. Labels starting with a colon are considered to be local labels – labels without the colon are considered to be global, and mark the outer limits of a local label's reference. This is extremely useful for all those throwaway labels I use while writing a large program, such as loop263, next22a, blah84, etc. Now each routine gets a global label, while internal loops get names like :loop or :next. It makes code much cleaner to look at, and lets the programmer think about more important problems at hand than what label name to use next.

The only feature I missed after a fairly long session of editing was a search command. Steve said he was working on adding that, and went one better with a bookmark feature that Merlin 128 has and he uses often.

Hitting F1 starts the assembler, and it's just a simple matter of sitting back and letting the computer do the work. If any errors are encountered, a fairly descriptive error message is generated – just hit a key and the cursor is sitting on the erroneous line, ready to be fixed. There are apparently a few bugs still lurking in the assembler (I didn't find any, but Steve has) and those will certainly be ironed out before the first official release.

Hitting F2 enters Jammon, where we can take a look at the code just generated, move memory around, and of course test the code. Hitting X will return you right back to the editor, with all your source code intact. In all this, not a single bit of disk access is necessary, since ECA makes use of the extra 64k of RAM in bank 1 of the SuperCPU as a temporary holding spot for the source code while Jammon is running.

Hitting F7 brings up a disk menu where source code can be loaded and saved. Of course, a directory and DOS command prompt are available here as well. I had no problem bringing source code into ECA from Turbo Macro Pro/Style by saving the source out as a SEQ file, and then converting the file into a PRG file with Jeff Jones's DirectoMeister III from Loadstar #156.

This program opens the door to development on the SuperCPU, and is bound to get at least a few people making some SuperCPU specific programs. As with all free programs, the best thing you can do is to tell the author that you like and use his program – keep the encouragement level up, and you'll be seeing many more exciting things on your computer!

Forgotten Shadow Register Contest!

Way back on issue #111, I offered to give away my DPS-1101 Printer: I have long since given the printer away to a local friend. To date *no one* has responded to this challenge. Anyone remember this?

JEFF'S CHALLENGE: SHADOW REGISTERS is designed to show you what you can do with raster interrupts. There's a flicker that we've deliberately left in the routine. All you have to do is write your own version of SHADOW REGISTERS, make it flicker free, and send it to LOADSTAR before October 1, 1993. I (Jeff) will pick the best one and send the author my own DPS-1101 daisy wheel printer and all the ribbon they can possibly use. The printer is noisy and slow, but has better text output than a laser printer. Just change it to device 5 and use your dot matrix as 4 for graphics and the DPS-1101 as 5 for correspondence. This is no big contest so there's no cash prize unless we decide to publish the

winning program. We'd prefer if people who are NEW to interrupts give it a try. We've given you everything you need to tackle this job and we'd like to see how it works.

Using Monitors To Step On Up To Machine Language

By Jeff Jones. A young man once called me and said that he just didn't get the whole ML thing no matter how hard he tried. The first ML code I'd ever written was a patch for one of Scott Elder's programs, then some routines for STAR CATALOGER. This stupendous code that took me a half hour to write, printed the message "Get Ready..." Egad!

I explained to him that machine language was easy to learn, but the thing that always stumped me for years was the actual assembler interface. Every assembler was THE BEST assembler for power programmers. When I bought my first assembler, five years ago, I had studied and studied machine language, and thought that I was ready to write in assembly language. Boy was I wrong! My first assembler was so confusing that I STILL can't use it.

I quickly grew frustrated and decided that I was just too stupid to write machine language programs. I concentrated my efforts on other things. Briefly -- just BRIEFLY, I considered writing machine language code in my EPYX FAST LOAD cartridge. The thought was tossed from my head like an ultimate sin. My machine language tutor books warned against using a mere monitor to write ML code. Even the EPYX documentation said that it would have been harder to do in a monitor. So I didn't try it.

Since then friendlier and more powerful assemblers have burst on the market such as Turbo Assembler, CheapASS, Pal, Buddy, Merlin, and Panther. Still I felt too stupid to program ML.

Then one day I needed a quick and simple routine so I asked Mike Maynard to write it for me. He said "Sure!" He powered up a C-128 and typed MONITOR. In the monitor mode, he wrote the routine in about thirty seconds. I watched on, amazed. It couldn't be possible. I understood everything he did. I could have DONE that myself!

It still took me another year or so to get up the courage to fire up another assembler. But before BUDDY, I had to take the plunge. The plunge for me was SUPER Snapshot's Code Inspector monitor. There I hacked out my first bytes of working machine language code. It was there that I saw that I wasn't stupid. There was nothing wrong with my head. I really COULD do it.

I don't want to change the rules. I'm not a guru, but if anyone asked me, I'd suggest a good monitor before graduating to a full-fledged assembler. For one thing, the monitor is more straightforward. You're writing the code right into the memory location where you want it, using mnemonics, just like an assembler. There are no linkers, offsets, pseudo-ops, source code, and assembler specific syntaxes to confuse you. Your power is EXTREMELY limited in a monitor, but you can still write good code and test it immediately -- just like BASIC! The monitor doesn't confuse you.

Best of all, the limitations of a monitor PREPARE you for the power of an assembler. You start saying things like "I wish I could relocate this code, or re-use that portion."

The graduation from monitor hacking to assembler programming was like predestination. I was prepared for all those powerful options. I even LONGED for them.

Programming in a monitor was the best thing that happened to my programming life. Not only am I using two assemblers now, and programming in assembly, but also even understand the C-64's architecture better. I also understand other languages better.

As a rule, I don't program in monitors anymore. *Now* they confuse me and I feel extremely restricted. But if a routine is short enough and unimportant enough (like a dedicated screen switcher), I'll write in a monitor. A full-featured monitor is an asset to a programmer. The assemble command isn't an obsolete tool.

It's kind of funny: I was talking to Doctor Teranna, the creator of the maze algorithm for MAZE MAN, published on issue #74. I asked him what type of assembler he used. I didn't recognize the name he mentioned, but when he described the

assembling process, he was describing a monitor, not today's assembler. Then I remembered that Dr. Teranna has been programming for decades. He remembers a time when you had to write out your assembler code on paper and then punch it into cards (no keyboard or CRT). So for him a monitor can be considered a powerful assembler, especially since his mind is so math-oriented.

Today's "decent" monitor really is yesterday's "full-featured" assembler. You probably have a monitor of some sort at home already. There's one in MAVERICK, EPYX FASTLOAD (though it's non-standard and confusing), WARP SPEED, SUPER AIDE, the C-128 (excellent), SUPER SNAPSHOT (the best monitor of all).

ACTION REPLAY has a monitor, but it ain't all that great. Plus it's slow and doesn't show you the ROMs.

If you want to experiment with ML, you probably don't have to drop one cent for an assembler until you're ready.

Here I go again touting SUPER SNAPSHOT. The reason why I say it's the best is because it provides the best environment for programming in a monitor. The monitor is friendly. You don't have to use spaces between commands and numbers. You are also not required to type leading zeroes. This means you can type M801 instead of M 0801 and get away with it. You can also use decimal if you want.

Another boon for SNAPSHOT is that all the other monitors mentioned, with the exception of EPYX, don't represent ALL characters when displaying memory. Instead they replace just about any character with the high bit set with reverse periods. NOT NICE. There is a character defined for every number between 0 and 255. Why not display them?

Don't buy SUPER SNAPSHOT specifically for programming in its monitor. You can get BUDDY for a lot less. BUDDY is recommended by me. Today's assembler IS more useful for programming than today's monitor but you have to have assembled before you understand them – A chicken before the egg paradox.

There is a down side to programming in monitors. That's the fact that you have a hard time editing what you've written. When you submit a program, we'd rather have source for *easy* changes.

Including Fonts, Sprite and ML within Your Abacus-Compiled Programs

By Jeff Jones. I see a lot of submitters using bank switching in order to use fonts with their compiled programs. Nothing is wrong with bank switching, but there IS an alternative. You CAN use bank 0 (the normal bank, and screen 4, (the normal screen) while using a font -- even if the compiled program is over 100 blocks long.

I do it using OPTION F, from BASIC 64's ADVANCED DEVELOPMENT PACKAGE, accessed through option 3 of the main menu. Option F is called "CODE-START", which defaults to 7577. This option is used to raise the start of the compiled code. Before I explain this, let me explain, in layman's terms, the two major sections of a compiled program:

A. The RUNTIME MODULE
B. The OBJECT CODE

A RUNTIME MODULE is KERNAL type code that object code relies upon in order to execute. Your compiled programs can't run without this module, which is about 19 blocks long.

OBJECT CODE is computerese for output file -- or a file that has been translated from one form into another. In the compiler's case, this translation is from BASIC code to PSEUDO code or P-code. This is why the object files have the prefix, "p-".

You may have noticed that a few compiled programs we've published obviously incorporate fonts and sprites but have no font files. This is because I've put the F function to good use.

Normally the P-code starts at 7577 decimal. With option F, you can raise (not lower) the start of P-code. Here's how it ties into fonts:

Because bank 0 ends at $3FFF and most compiled programs end way after that, fonts can't be placed after a compiled program without changing banks. Couple this with the fact that compiled programs must be LOADED and RUN from $0801 and you can see why most people opt to go ahead and switch banks.

My favorite location for a font is $2000 (8192). A font LOADed there ends at $27FF (10239). I just use option F to raise the start of P-code to 10240. This will generate a different type of compiled program:

A. RUNTIME MODULE 2049-7576
B. A SEA OF ZEROES 7577-10239
C. COMPILED PROGRAM 10240 to end

The program will be a few blocks longer than a program compiled normally.

This configuration allows you the unique opportunity to insert a font at 8192 ($2000) -- WITHIN the program itself. Of course your program should respect this font with the following POKE:

POKE 53272,24

Inserting your font at $2000 is best done with a monitor in five easy steps:

1. LOAD compiled program
2. Enter ML monitor
3. BLOAD font at $2000
4. Exit monitor
5. SAVE altered program.

LOADing the font in the monitor doesn't affect BASIC pointers, allowing you to SAVE the program normally, without having to know where the end of the program is.

If you don't have an ML monitor:

```
1. LOAD compiled program
2. SYS57812"font name",8,0:
POKE780,0
   POKE781,0:POKE782,32:
SYS65493
3. SAVE altered program
```

The new file you've just created contains your compiled program with a font sandwiched in the middle. And guess what! You have room between 7577 and 8192 for ML or data. However, since sprite data can't be seen between $1000 (4096) and $2000 (8192) by the VIC chip, if you store sprite data there, you'll have to copy it elsewhere to use it. Remember, though, you can use F to raise the start of p-code even farther to accommodate another font, sprites, hi-res screens, etc.

The beauty of this is that you have your program, sprites, fonts, graphics, and ML all in one file. This shortens LOADing time and gives you a neat package.

There's no need to protect your graphics and ML from BASIC when they reside safely inside the compiled program. This memory is untouched unless you deliberately POKE it or somehow lower the bottom of BASIC to within this pool.

If you Starlink data beyond the end of the program, you must protect that memory with the E option. When run, the compiler physically clears *all* high memory to zeros. The compiled CLR command also does this. Nothing escapes. So if you're linking data at 40960 and higher, make your highest memory location used 40959.

Top 15 New Slogans for the Democratic Party

☺ **15.** Okay, he's a hound -- But he's OUR hound!
☺ **14.** Vote for Our Guy or These Dole Viagra Pictures Hit the Internet
☺ **13.** If the Dome is A-Rockin', Don't Come A-Knockin'!
☺ **12.** When the Going Gets Tough, We Bomb Iraq.
☺ **11.** So Spank Us!
☺ **10.** It's Not Our Fault All the Good Ones Get Shot
☺ **9.** Our Pants May Fall, But Your 401 (K) Value Won't!
☺ **8.** Felonies Dismissed While You Wait
☺ **7.** Slide your fine ass over here and give us a vote, Darlin'!!
☺ **6.** Mmmmmmm... Peachy!
☺ **5.** Oh, So You Wanna Play Rough, Huh?
☺ **4.** C'mon -- We Didn't Know He Was *THAT* Horny!
☺ **3.** We've Got Cigarfignugen!
☺ **2.** Impeach THIS!!!
☺ and Top5's Number 1 New Slogan for the Democratic Party...
☺ **1.** Laid in America

[This list copyright 1998 by Chris White] [The Top 5 List top5@gmbweb.com http://www.topfive.com] [To forward or repost, please include this section.] [You like to receive credit for your work, and so do we.]

Top 15 New Slogans for the Republican Party

☺ **15.** Putting the HIP back in Hypocrisy
☺ **14.** Adultery with Dignity
☺ **13.** $40 million is peanuts compared to what *they* want to give poor folks!
☺ **12.** Moraler Than Thou!
☺ **11.** Let's Get Ready To Stumblllllllllllllllllllllllllllllllllle!
☺ **10.** Please, don't squeeze the Chairman
☺ **9.** 3% More Ethical Than the Other Guys
☺ **8.** We Love to Pry, and it Shows
☺ **7.** 1000 Points of Spite
☺ **6.** LALALALA! We're Not Listening!!!!
☺ **5.** Your choice, America: Vote Republican or roast in hell.
☺ **4.** The New GOP -- Now with Black folks!
☺ **3.** Upholding Principle and Truth Since 1998
☺ **2.** With our finger on the pulse of the American peop... Strom? Strom?!
☺ and Top5's Number 1 New Slogan for the Republican Party...
☺ **1.** Impeachment: Because the Secret Service Won't Let Us Get Close Enough To Lynch

LOADSTAR LETTER #65
J&F PUBLISHING 606 COMMON STREET SHREVEPORT LA 71101
- COMMODORE NEWS
- COMMODORE VIEWS

Bulk Rate
U.S. Postage
PAID
Shreveport LA
PERMIT #85